Introduction:

Expeditious urbanization is leading to smarter cities with the intention to provide core infrastructure and a decent quality of life to its citizens, a clean and sustainable environment. Smart City is a wave of the future with few countries already at its crests with statistics expecting 60% of the population living in cities by 2050. Smartness can be established in any sector of the urban area: be it in an office, school, transportation infrastructure, library etc.

Our idea is centralized to a library with focus on energy consumption, security, safety and comfort of its occupants. The sensors provide the current information about the temperature, humidity, light, motion in the library. The raw data is transformed into context data by the input drivers. Energy consumption is controlled prudently by handling the lighting system, HVAC and other electronic components dynamically after performing modification on the context data. Security is obtained by the alarming system in case of unexpected and malicious entry into the environment under consideration during closed hours (12 am to 6am).

For the purpose of storing the data, we are using a relational database management system, SQLite3. The database is embedded into our system thereby making the SQLite suitable for our application.

Coming section describes the querying of data in the SQLite database using Python:

1. A connection is established to the database by creating a Connection object
2. A cursor object is created using the cursor method of the Connection object
3. Insert statement is used to add elements to the database
4. Select statement is executed
5. The fetchall() or fetchone() method is called depending on the action required of the cursor to fetch the data
6. After accessing the database, close the cursor and the database by using the close method for the cursor and connection object

Our database stores data in the form of a table with its contents as Date stamp, Topic and Sensor data. The mentioned procedure is followed to insert elements. The data to be stored is received via MQTT. The database script subscribes to two topics, namely SmartCities and Database. SmartCities topic corresponds to the data from the sensor script and Database topic to the actuator script. At the time of reception, the time is calculated in the format “YYYY-MM-DD HH-MM-SS”,using the datetime library. The topic and the corresponding data is stored. There is always a possibility that the data from the sensors and data that drive the actuator may repeat, this leads to redundant data to be stored. To cope with this issue, before storing the latest data stored is fetched from the database using the fetchone() method. This data is compared with the current value and if there exists a change, this new value is stored else it is discarded.

The database that is created is stored as a file with .db extension which can be viewed using the <https://inloop.github.io/sqlite-viewer/> .

The real time data is displayed using a graphical user interface, Tkinter provided by Python. Tkinter is a python interface to the Tk GUI toolkit. Creating a GUI using TKinter is done by first importing the TKinter module , secondly creating a main window, add necessary widgets to the main window and execute any event triggered action.

The main window is titled Smart Library, Group 25. The widgets used in the system are labels, buttons, optionmenu and checkboxes. The following values are displayed : current temperature, target temperature, library status, people entered, exited and current people count in the library, heater status, cooler status, light intensity, status of LEDs, humidity, humidifier status and dehumidifier status. An option for plotting the data stored in the database is provided for temperature, humidity, people count, heater, cooler, people count, light intensity, LEDs status’ with (x,y) axis as (date stamp, data). Matplotlib library is used for plotting the graphs. A Buzzer Override button is provided to the user to manually turn off the buzzer in situations of unauthorized personnel during the closed hours. On the click of this button, mqtt is used to publish this information to the output script thereby turning off the buzzer. This GUI retrieves data from the database that was created and displays real time data at the interval of 2s (can be varied).

Conclusion and discussion:

Our system is tested successfully for light control, HVAC control by turning on and off the plugwise circles , buzzer operation in case of intruders during the closed hours. The assumptions made are as follows. The HVAC system considered is a simple, basic on/off control wherein if the temperature is below the set point, cooler is turned on and if the temperature rises above the set point, heater is turned on. The scope of our implementation is not universally applicable to all the HVAC systems. Second assumption is the raise in the temperature of the room depending on the number of people in the room; a hard coded value is used for this purpose while in the real time scenario this is a trivial approach. To cope up in this situation, calibrations are needed to understand the temperature raise for a group of people. Another similar assumption as previous is the temperature increase due to the lights. With our implementation, there exists energy conservation when considering the existing system in the library. However with the cons mentioned previously, our implementation is a crude one and definitely needs extensions to meet the real time application.